Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

PUNCH 4NFDI

Foundation models for HEP

AISSAI worskhop on heterogeneous data and large representation models in science
2024-09-30

Anna Hallin

anna.hallin@uni-hamburg.de

AISSAI
AI for science, science for AI

Outline

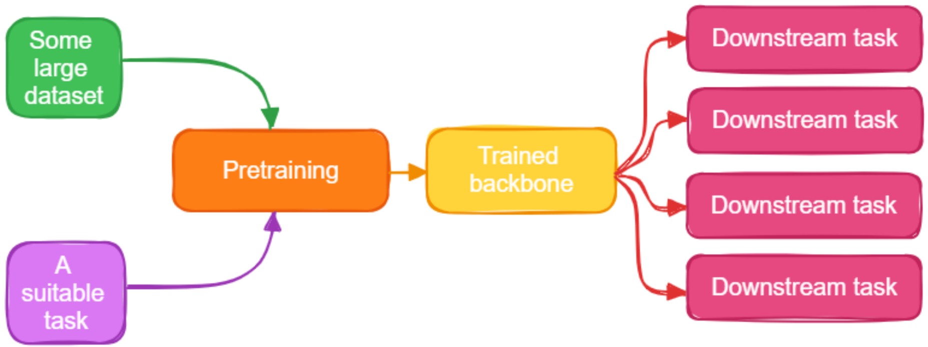
- Introduction to foundation models
- Foundation models in HEP
- A closer look at a foundation model for jet physics
- Outlook

Introduction to foundation models

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

What are foundation models?

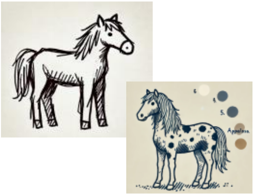
- **Pre-trained** on a certain (large) dataset for a certain task, **fine-tuned** to perform on a different dataset or a different task
- Better **performance** than training the downstream task from scratch



Why does it work?

- During pretraining, the model learns **aspects of the data** that are **useful** for downstream tasks
- The model has a "head start" compared to a model that needs to train from scratch

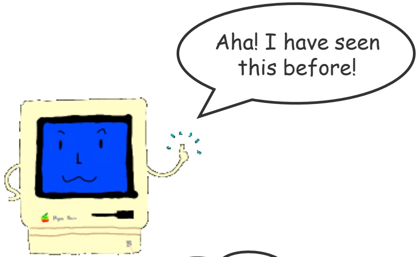
Pretraining



"Draw some of these animals"

Downstream task

"Which one of these is a horse?"



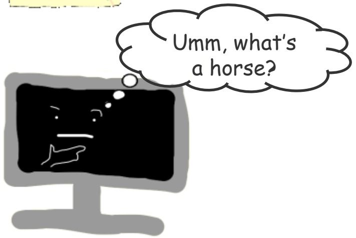
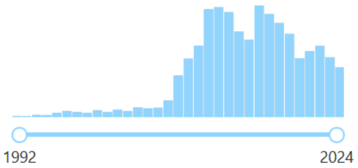
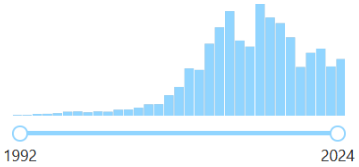


Image credits:
DALL-E
themarketingblog.co.uk
drawception.com

Benefits

- Foundation models may be expensive to train, but once pre-trained, downstream tasks require **less resources**
  - Human resources
  - Compute resources
- Can leverage the pretraining to **boost performance on small datasets**
- **Sharing** pre-trained models can provide others with access to resources that are normally not accessible for them (data, computing resources)





ATLAS and CMS: 18k papers
2024 so far: 696 papers

Examples of foundation models: language

- GPT-3 [1]
    - Input: text
    - Pretraining: next token prediction text generation (transformer)
    - **Finetuning**: conversational data + reinforcement learning with human feedback
      → ChatGPT

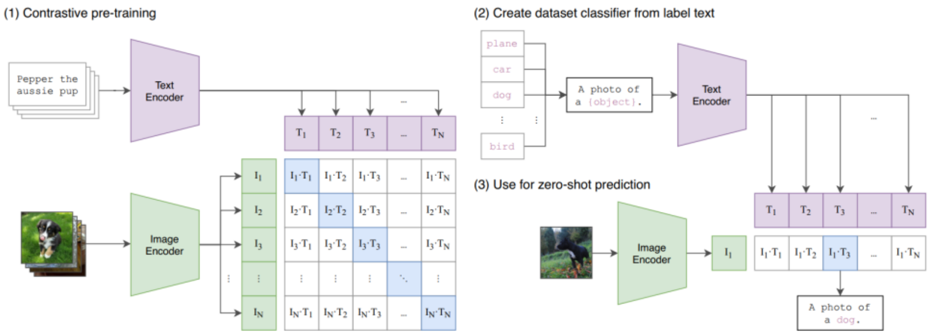
what are you?

I am ChatGPT, an AI language model developed by OpenAI. My purpose is to understand and respond to text-based inputs, helping answer questions, provide information, assist with tasks, or just have a conversation. I use patterns in the data I was trained on to generate meaningful and relevant responses to various prompts. What would you like to know or talk about?

[1] Brown et al, *Language Models are Few-Shot Learners*. arXiv 2005.14165

Examples of foundation models: images

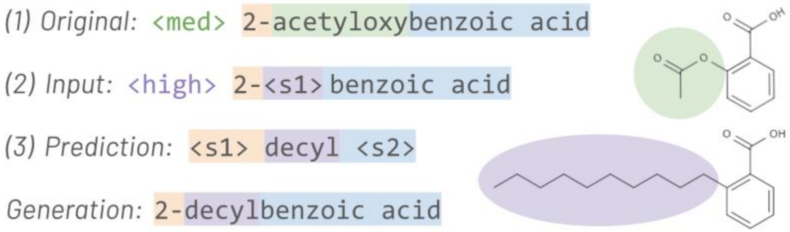
- CLIP [2]
  - Input: text and images
  - Pretraining: match images with descriptions (transformer for text, ResNet/ViT for images)
  - **Zero shot**: image classification



[2] Radford et al, *Learning Transferable Visual Models From Natural Language Supervision*. arXiv 2103.00020

Examples of foundation models: chemistry

- C5T5 [3]
  - Input: IUPAC names (standardized molecular naming system) + molecular property values
  - Pretraining goal: predict masked out token (transformer)
  - **Zero-shot**: Molecular replacement to change the molecule’s properties



Increasing a molecule’s octanol-water partition coefficient

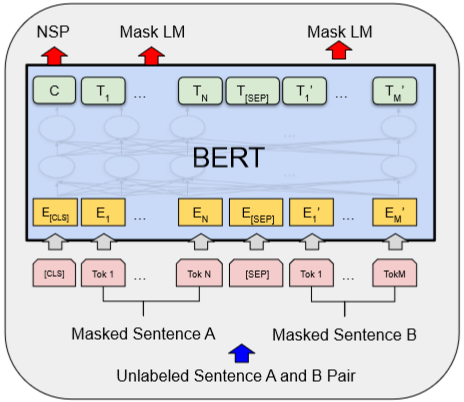
[3] Rothchild et al, *C5T5: Controllable Generation of Organic Molecules with Transformers*. arXiv 2108.10307

A note on transformers

- A transformer in itself is not a foundation model
- Foundation models do not necessarily need to be built on transformers

Pretraining

- Can be useful in itself, or a **surrogate task**
- Example of surrogate tasks: BERT [4]
  - **Masked language modeling** in addition to **next sentence prediction**
  - Masking out tokens allows bidirectional training: sees both previous and future words in order to capture the **context within a sentence**
  - Next sentence prediction captures **context between sentences**: does sentence B follow sentence A?



1810.04805

[4] Devlin et al, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv 1810.04805

Scale

Foundation models become powerful because of **scale**: **data**, **architecture**, **compute**

- Example GPT-3: 300B tokens, 175 billion parameters, estimated thousands of GPUs trained over several weeks ($\sim 10^{23}$ flops)
- Parameter scale example Parti (Pathways Autoregressive Text-to-Image model) [5]:

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

2206.10789

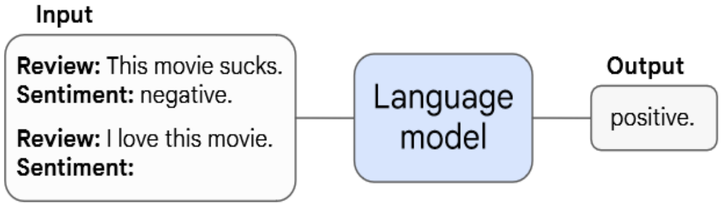
[5] Yu et al, *Scaling Autoregressive Models for Content-Rich Text-to-Image Generation*. arXiv 2206.10789

Emergent properties

A foundation model might be able to perform tasks that it was **not trained for**, and that were **not anticipated**. This behavior comes with **scale** [6].

Examples for a natural language model only trained to generate text:

- Translation
- Coding
- Basic arithmetic
- Sentiment analysis
- Few-shot and zero-shot learning



2206.07682

[6] Bommasani et al, *On the Opportunities and Risks of Foundation Models*. arXiv 2108.07258

Scale: when to stop?

Can you **predict the performance** of a larger model, without having to train it?

Maybe! In the context of language models (autoregressive transformers), it has been shown [7] that the cross-entropy **loss improves with scale** according to simple power laws.

- Dependence on number of **parameters** N if you have unlimited data and unlimited compute:

$$L(N) = \left(\frac{\mathrm{N_c}}{\mathrm{N}}\right)^{\alpha_\mathrm{N}}, \qquad \alpha_N \sim 0.076, \qquad N_C \sim 8.8 \times 10^{13}$$

- Dependence on number of **parameters** N and **data** D given an early stopping criteria for compute:

$$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}, \qquad \alpha_D \sim 0.095, \qquad D_c \sim 5.4 \times 10^{13}$$

[7] Kaplan et al, *Scaling Laws for Neural Language Models*. arXiv 2001.08361

Foundation models for HEP

Approaches to foundation models in physics

- **Use** large language models for applictions in physics
  - **ChATLAS**: an AI assisant in development for the ATLAS experiment at CERN, to better utilize knowledge currently dispersed across a large variety of documents
- **Teach/adapt** large languge models to do maths and physics
  - Symbolic maths: compute integrals and solve differential equations by treating equations and their solutions as a **translation** task [8]
  - Number embedding in text: treat numbers as a **different entity** than text, to allow the model to ”understand” numbers [9]
- Take **inspiration** from large languge models and others, **build from scratch**
  - The remainder of the talk will focus on this approach

[8] Lample and Charton, *Deep Learning for Symbolic Mathematics*. arXiv 1912.01412
[9] Golkar et al, *xVal: A Continuous Number Encoding for Large Language Models*. arXiv 2310.02989

Natural language vs high energy physics

Text

- Characters, (sub)words, symbols…
- Order matters
- Meaning builds across many sentences

HEP

- (Mostly) continuous numbers
  - Single numbers
  - Sets of numbers (vectors, time series)
- Can be permutation invariant
- Some sets of numbers like 4-vectors carry special meaning
- Symmetries might be present

A particle physics foundation model example

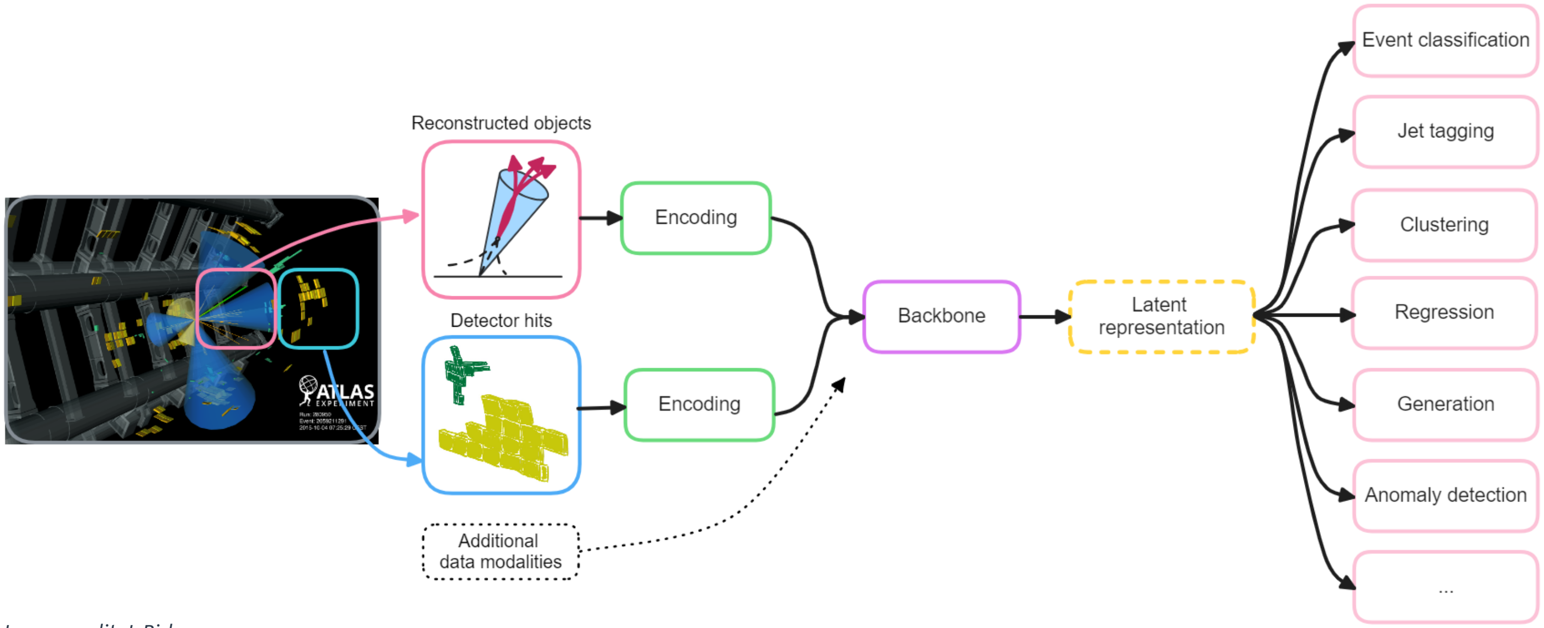


Image credit: J. Birk

A selection of foundation models for particle jets

Jets are important and common objects in particle colliders. They consist of a **collimated spray of particles** (constituents), which originate from the decay of a particle in the detector.

- ParticleTransformer (ParT) [10]
- Masked particle modeling (MPM) [11]
- OmniJet-α [12]
- OmniLearn [13]

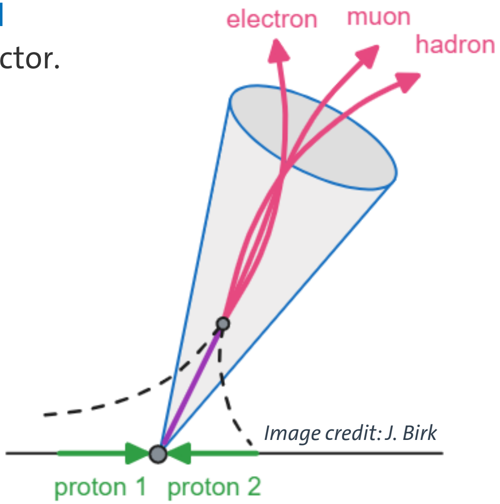


Image credit: J. Birk

[10] Qu et al, *Particle Transformer for Jet Tagging*. arXiv 2202.03772
[11] Golling et al, *Towards Self-Supervised High Energy Physics Foundation Models*. arXiv 2401.13537
[12] Birk, **AH**, Kasieczka, *OmniJet-α : The first cross-task foundation model for particle physics*. arXiv 2403.05618
[13] Mikuni and Nachman, *OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks*. arXiv 2404.16091

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)
OmniJet-α	Next token prediction (generation)	Transformer	Cross-entropy next token prediction	Classification (tagging), Generation (unconditional)

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)
OmniJet-α	Next token prediction (generation)	Transformer	Cross-entropy next token prediction	Classification (tagging), Generation (unconditional)
OmniLearn	Generation + classification	Transformer + diffusion	Cross-entropy class labels + diffusion velocity parameter	Classification (tagging: different dataset, different experiment, different collision type; anomaly detection), Generation (conditional), Reweighting and unfolding

Tokenization for generative tasks

- **Language models** need to **turn text into numbers** (which is what our models can work with), use tokenization: text → sequence of integer tokens
- In physics, we already have numbers, but our **architecture** can force us to **tokenize**:
  - Regression loss – no tokens needed, but has so far seemed to be more difficult
  - Cross-entropy loss – powerful, but need discrete numbers = tokens
- Example of a particle jet:
  - $\text{Jet} = \{p_1, p_2, \dots, p_N\}$
  - $p_i = \{p_T, \eta, \phi, \text{PID}, \text{charge}, \dots\} \rightarrow \text{token}_i$
  - Jets as **sequences of integers**:
    $\{<\text{start token}>, \text{token}_1, \text{token}_2, \dots, \text{token}_\text{N}, <\text{stop token}>\}$

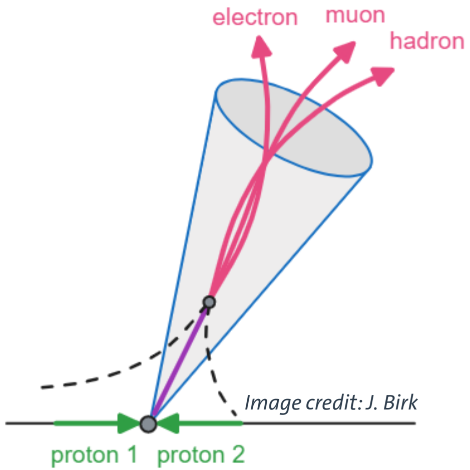
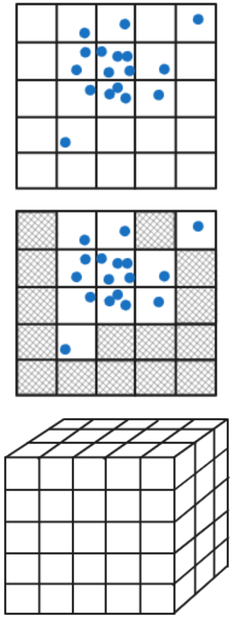


Image credit: J. Birk

Binning

- Divide each dimension into bins
- Sub-optimal **coverage**
- **Vocab size** becomes $\prod_{i \in features} n_{bins,i}$
  - Tokens → Embedding: `Linear` $(n_{\text{tokens}}, d_{\text{embed}})$
  - Embedding → Tokens: `Linear` $(d_{\text{embed}}, n_{\text{tokens}})$
  - Example: 100 000 tokens with embedding dimension 128 → 25.6M parameters



Vector Quantized Variational Autoencoder (VQ-VAE) [14, 15]

Learns an **embedding space** that gives the best reconstruction; less sensitive to adding dimesions

- Unconditional tokens: tokenize one constituent at a time, **1:1 correspondence**
- Conditional tokens: sees all constituents, adapts the tokens → one token can **cover multiple parts** of feature space

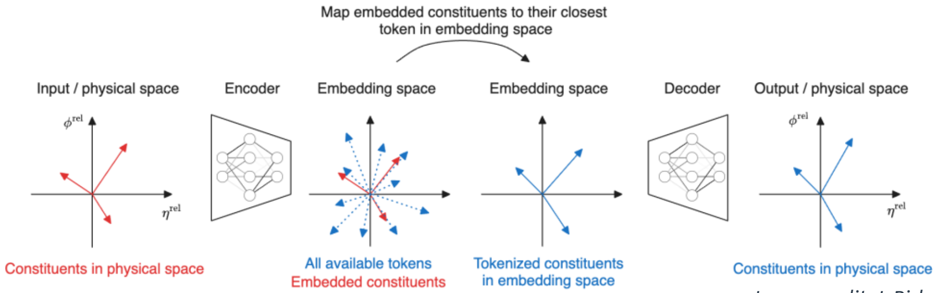
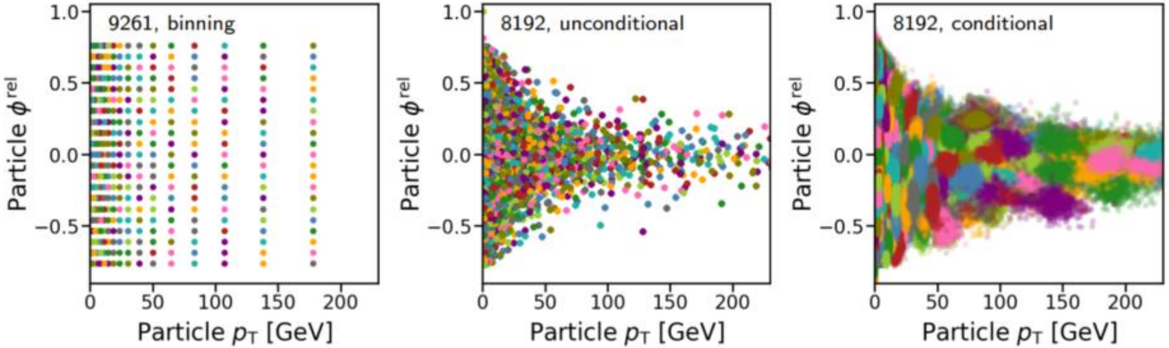


Image credit: J. Birk

[14] van den Oord et al, *Neural Discrete Representation Learning*. arXiv 1711.00937
[15] Huh et al, *Straightening Out the Straight-Through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks*. arXiv 2305.08842

Binning vs VQ-VAE

- VQ-VAE adapts to the shape of the data
- Conditional tokenization covers more of the phase space

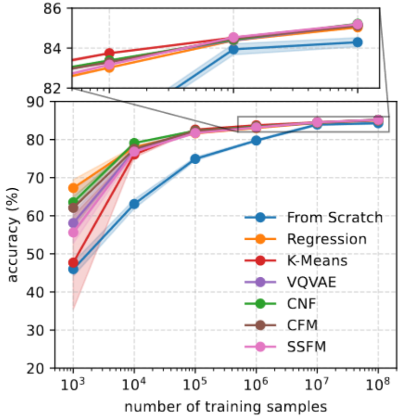


2403.05618

Do we really need tokenization?

- In a new paper on MPM [16], various reconstruction tasks for the pretraining have been tested, including tasks not requiring tokenization.
- Downstream tasks such as classification, weakly supervised anomaly detection, second vertex finding and heavy track identification seem to work well with continous pretraining.

Classification performance, from 2409.12589



[16] Leigh et al, Is Tokenization Needed for Masked Particle Modelling? arXiv 2409.12589

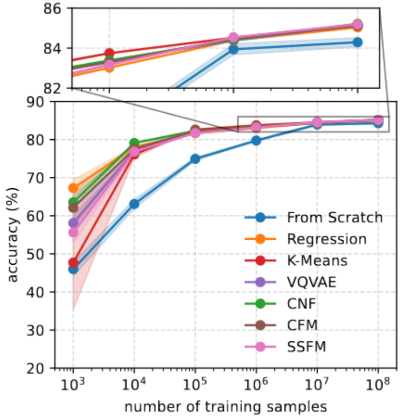
Do we really need tokenization? It depends on what you want to do!

- In a new paper on MPM [16], various reconstruction tasks for the pretraining have been tested, including tasks not requiring tokenization.
- Downstream tasks such as classification, weakly supervised anomaly detection, second vertex finding and heavy track identification seem to work well with continous pretraining.

So, **do we need tokens**?

- For this specific pretraining target and these downstream tasks, it seems to not be needed.
- For an autoregressive generative model that can learn the number of constituents of a jet from context (eg. OmniJet-α), it is currently still needed.

Classification performance, from 2409.12589

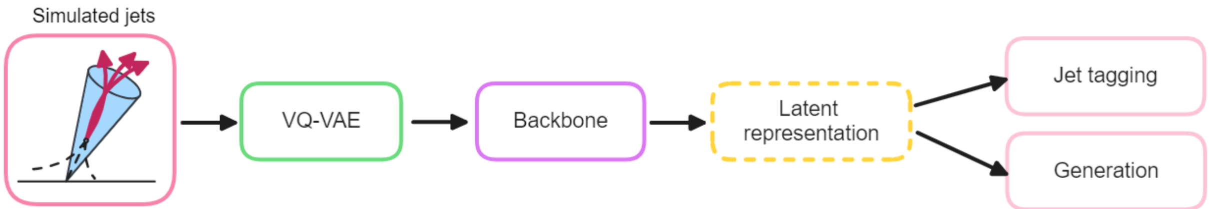


[16] Leigh et al, Is Tokenization Needed for Masked Particle Modelling? arXiv 2409.12589

A closer look at OmniJet-α

A closer look at OmniJet-α

- OmniJet-α is the first foundation model for particle physics that is able to **task-switch**: unsupervised **full jet generation** and supervised **classification**
- **JetClass dataset** [17] with **10M jets of each type**, originally used in ParT
- Tokenizes with **VQ-VAE**
- Uses a transformer for **generative pretraining** based on the GPT-1 architecture [18] with **next-token-prediction** as training target. $p(x_j | x_{j-1}, \ldots, x_1, <\text{start token}>)$



[17] http://dx.doi.org/10.5281/zenodo.6619767
[18] Radford et al, *Improving language understanding by generative pre-training*. 2018.

Generation

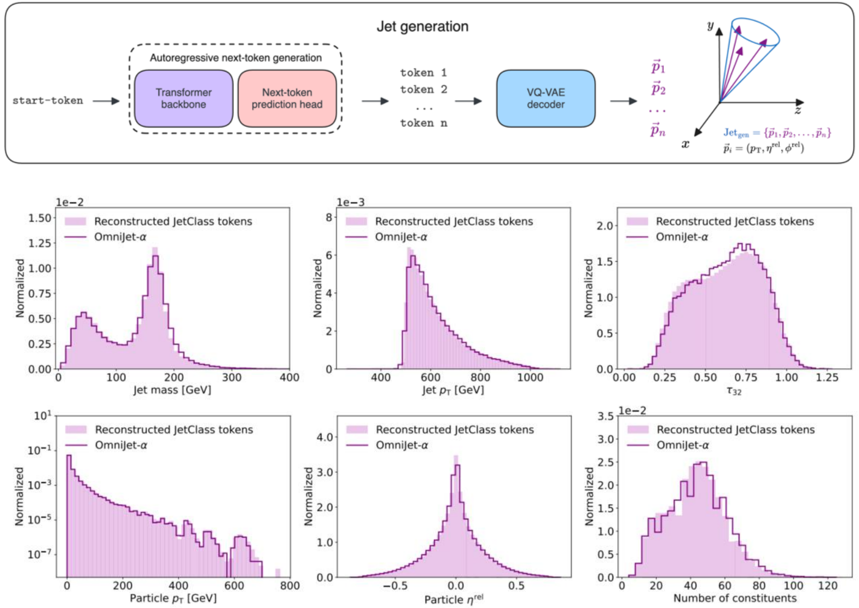
During generation, the model generates tokens **auto-regressively**:

- Model has learned $p(x_j | x_{j-1}, \ldots, x_1, <\text{start token}>)$

- Model recieves `<start token>` and generates until it generates a `<stop token>` or the maximum sequence length is reached

Generally **good agreement** to truth distribution

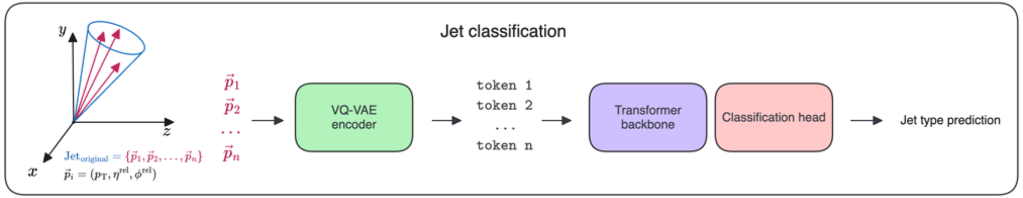
Constituent p_T spectrum tail has few events $\rightarrow$ the limited codebook size shows up as bumps



Transfer learning: classify quark/gluon vs hadronic top jets

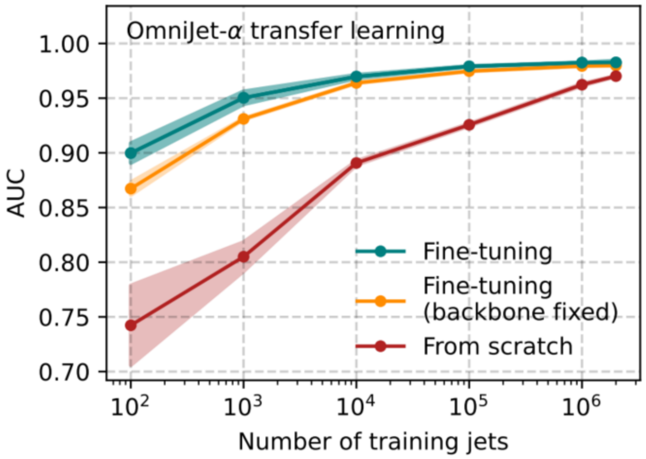
The next-token-prediction head is changed to a classification head. We tested three approaches:

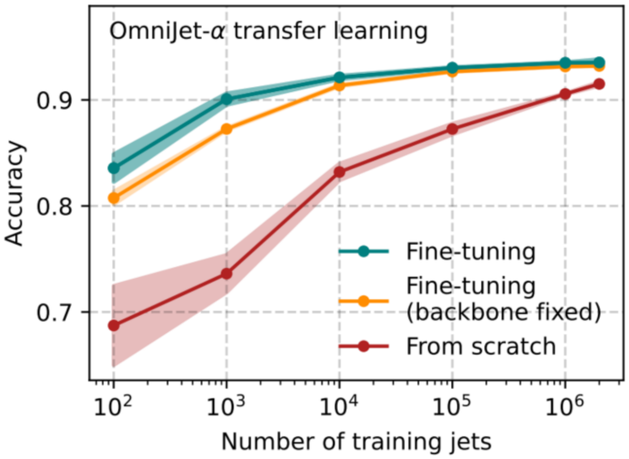
- **From scratch**: all weights are initialized from scratch, no pre-training is used
- Fine-tuning: load weights of the pre-trained generative model
    - regular **fine-tuning**: all weigths can change
    - **backbone fixed**: weights of the pre-trained transformer backbone are held fixed



Transfer learning results

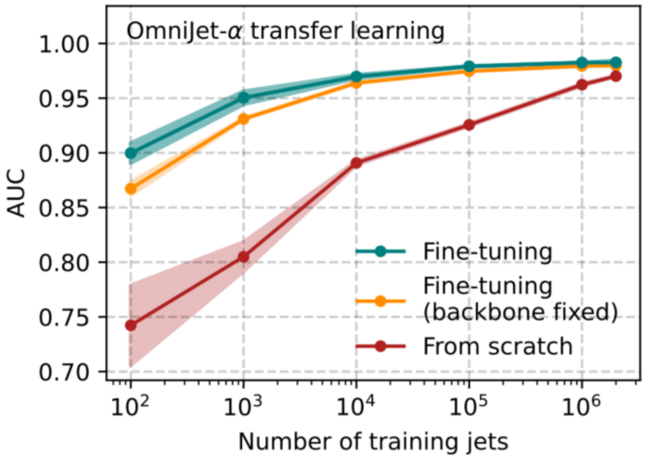
- Significantly better result when using pre-training
- Full fine-tuning slightly better than backbone fixed

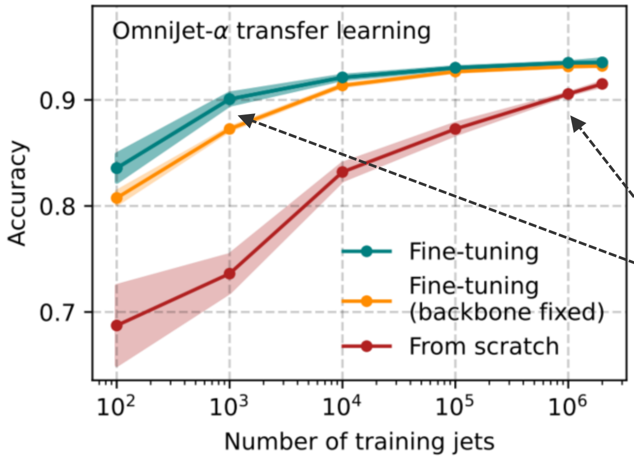




Transfer learning results

- Significantly better result when using pre-training
- Full fine-tuning slightly better than backbone fixed





Pre-trained model requires only 1000 training events to reach the same accuracy level that the "from scratch" model reaches with 1M events.

Outlook

Creating your first foundation model

- Downstream tasks
- Pretraining
    - Training goal
    - Architecture
    - Loss
    - Tokenization or not
    - Unsupervised, self-supervised, supervised...
- Input data
    - Multi-modal? Why and how?
    - Add physics info? Constraints, symmetries...

Conclusion and outlook

- Foundation models are **multi-task** and **multi-dataset** machine learning models that once **pretrained** can be applied to a variety of **downstream tasks**
- The successful development of foundation models for physics would be a **major breakthrough**, improving performance and saving human and compute resources
- Open questions:
    - What is the most efficient **representation** of the data?
    - How to introduce **multi-modal** data?
    - Exploring architectures and **pretraining strategies**
    - Expanding to further **downstream tasks**
    - Investigating effects of **scaling**

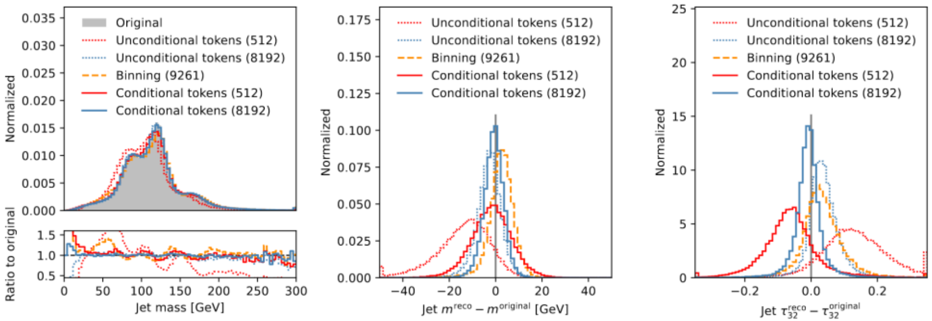
Backup

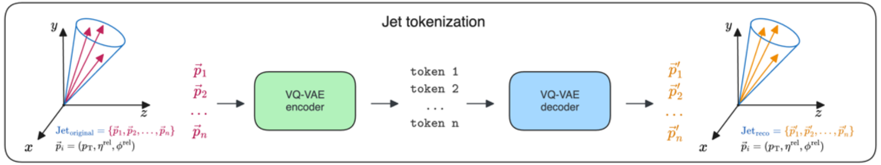
Tokenization

Compared several approaches:

- Binning
- VQ-VAE
  - Unconditional
  - Conditional
  - Different codebook sizes (vocab sizes)

We proceed with **conditional tokens** with codebook size **8192**.



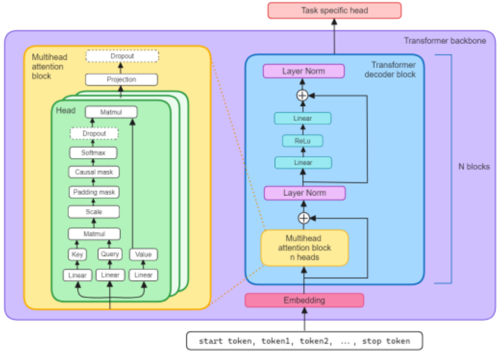


Backbone training

The transformer backbone is trained with the **next-token-prediction** head.

- **Causal mask** prevents attention to future tokens
- n heads = 8, N GPT blocks = 3 results in 6.7M parameters
- Model learns to predict the next token, given a sequence of previous tokens: $p(x_j | x_{j-1}, \dots, x_1, < \textbf{start token} >)$

0				
0	153			
0	153	5489		
0	153	5489	51	
0	153	5489	51	8193

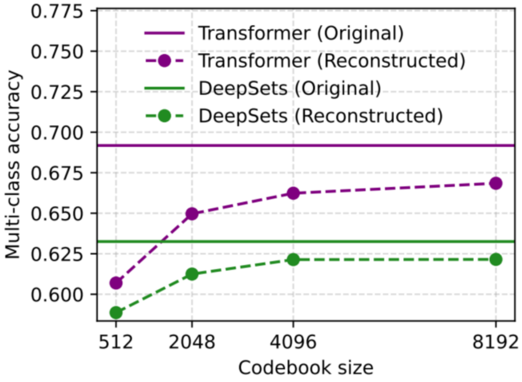


Dataset

- JetClass: 10 classes of simulated jets with **10M jets of each type**, originally used in ParT
- **Tokenize all 10 classes** at once to evaluate tokenization performance
- For pretraining: use **10M *q/g* jets and 10M $t \rightarrow bqq'$ jets**.
- No class labels are passed to the model during pretraining.
- Use **constituent features** p_T, η^{rel}, φ^{rel} (rel = relative to the jet axis), no jet-level information

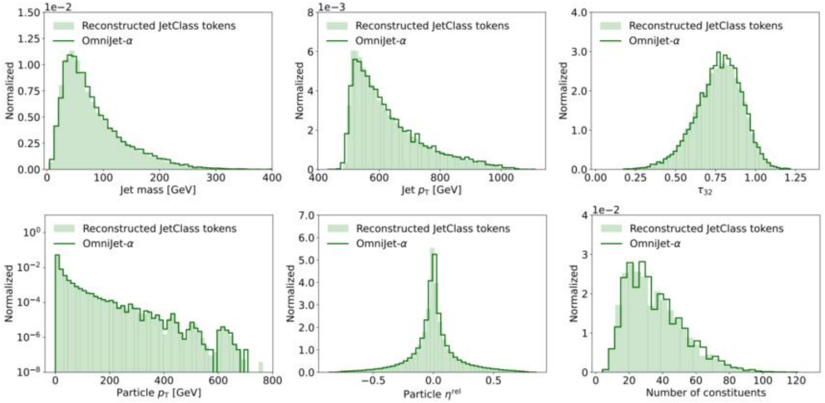
Quantifying tokenization information loss in OmniJet-α

- Train a **multi-class classifier** on all 10 classes of JetClass (note: this is not a reconstructed vs truth test)
- Two types of classifiers are tested: **transformer** and **Deep sets**
- Train on original JetClass data to obtain an **upper limit**
- Accuracy starts **plateauing** at a codebook size of 8192

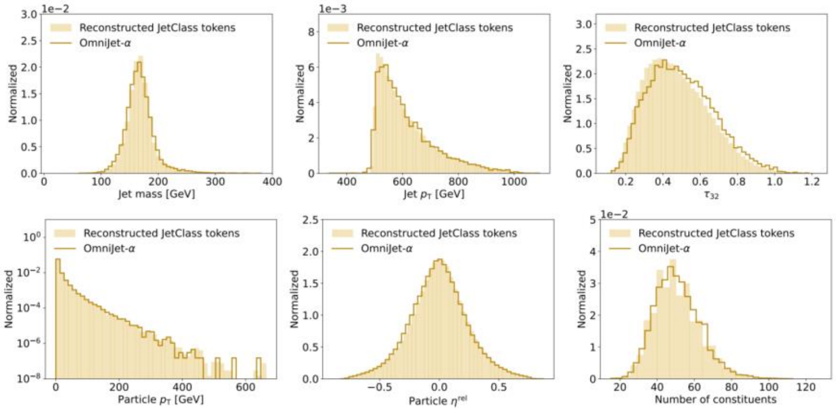


Generative results, single-jet type training

- *q/g* jets

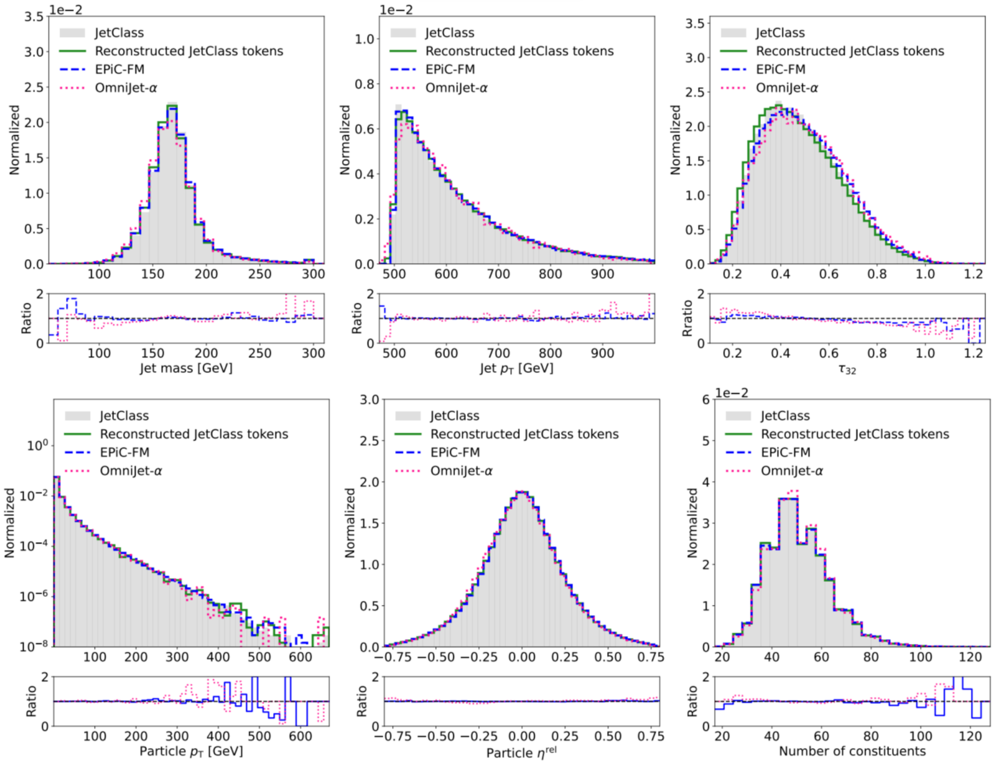


- $t \rightarrow bqq'$ jets



Comparison of generative capabilities, $t \rightarrow bqq'$

- EPiC-FM [19]: flow matching, **no tokenization**
- Ratios compare OmniJet-α and EPiC-FM (kinematics version) to **their respective truths**
- **Both** models are **doing well**
- OmniJet-α has a slightly higher discrepancy in the tails, except for constituent η^{rel} and number of constituents



[19] Birk et al, *Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information*. arXiv 2312.00123.